Une exploration de l'architecture des réseaux de neurones pour la modélisation de la compositionnalité sémantique

Chloé Cimpello

Une exploration de l'architecture des réseaux de neurones pour la modélisation de la compositionnalité sémantique

_ Chloé Cimpello _

UFR LLASIC

Mémoire de master 2 professionnel - 20 crédits

Mention : Sciences du langage - Spécialité : Industries de la langue - Parcours : TALEP

Sous la direction de : Tim Van de Cruys et Thomas Lebarbé

Laboratoire : Institut de Recherche en Informatique de Toulouse - Équipe MELODI

Année universitaire 2014-2015

Une exploration de l'architecture des réseaux de neurones pour la modélisation de la compositionnalité sémantique

_ Chloé Cimpello _

UFR LLASIC

Mémoire de master 2 professionnel - 20 crédits

Mention : Sciences du langage - Spécialité : Industries de la langue - Parcours : TALEP

Sous la direction de : Tim Van de Cruys et Thomas Lebarbé

Laboratoire : Institut de Recherche en Informatique de Toulouse - Équipe MELODI

Année universitaire 2014-2015

Remerciements

Merci.

Mots-clefs : linguistique, informatique, traitement automatique des langues, réseaux de neurones, compositionnalité, sémantique, vecteurs, apprentissage automatique

RESUME

Ce mémoire présente une évaluation du modèle de réseau de neurones, appelé autoencodeur, qui permet de capturer le sens de couples adjectif-nom en anglais. Ce modèle fonctionne sur la base de la représentation du sens des mots par un vecteur contenant les indices des lemmes constituant le contexte des mots en question. Ces indices sont attribués aux lemmes en fonction de leur fréquence dans notre corpus issu de *Wikipédia*. Notre modèle est évalué sur un test de similarité entre deux couples adjectif-nom puis sur un test de recomposition des vecteurs de contexte du couple adjectif-nom à partir des vecteurs de contexte de ses composants pris séparément. Les résultats de ces deux tâches est ensuite comparé aux modèles déjà existants suivants : l'addition de vecteurs (modèle additif), le modèle additif pondéré avec un coefficient plus fort sur le vecteur du nom, le modèle basique où seul le vecteur contexte du nom est pris en compte, et la multiplication de vecteurs (modèle multiplicatif).

Keywords : linguistics, computer sciences, computer linguistics, neural networks, compositionality, sémantics, vectors, machine learning

ABSTRACT

This dissertation presents an evaluation of a neural network model called autoencoder in order to capture the meaning of adjective-noun couples in English. This model works on the representation of words meaning by a vector countaining the index of the words' context lemmas. These indexes are assigned to lemmas according to their frenquency in our *Wikipedia*-extracted corpus. Our model is evaluated on similarity task between two adjective-noun couples, and then on a task of recomposition of adjective-noun couples vector from their separated components context vectors. These two task results were eventually compared to the following already existing models : the vectors sum (additive model), the weighted additive with a stronger rating on the noun vector, the baseline model where only the nouns vector is taken, and the multiplicative model (multiplication of vectors).

Table des matières

Introduction

L'entreprise et l'équipe MELODI

Dans le cadre de ma dernière année de Master professionnel Industries de la Langues, j'ai effectué un stage de 5 mois à l'Institut de Recherche en Informatique de Toulouse (IRIT), au sein de l'équipe MELODI, sous la tutelle de Tim Van De Cruys, chercheur au CNRS. Ce laboratoire de recherche en collaboration avec les universités de Toulouse ainsi que de nombreuses autres structures emploie 700 membres (permanents et non-permanents) répartis en 20 équipes différentes. L'équipe MELODI, qui a abrité mon stage est spécialisée dans deux domaines :

La représentation des connaissances :

- les méthodes et les principes de construction d'ontologies et la gestion de leurs évolutions
- la modélisation et la représentation de connaissances à l'aide de graphes ou de la logique

Le traitement de la langue naturelle:

- l'analyse et la description conceptuelle ou formelle de phénomènes linguistiques au niveau de la sémantique lexicale, de la phrase, du discours ou du texte
- l'apprentissage automatique de représentations structurées pour le TAL

C'est dans ce dernier domaine que j'ai été employée afin de travailler à l'évaluation et à l'exploration de l'utilisation d'un modèle de réseaux de neurones pour traiter la compositionnalité. Ce stage s'inscrit dans le projet Italodisco.

Italodisco

Le projet Italodisco (*Innovative Techniques for the Advanced Learning Of Distributional Compositionality*) a pour but de modéliser la compositionnalité sémantique de manière automatique et non-supervisée. Au cours des dernières années, de grands progrès ont été faits dans l'extraction de la sémantique de mots isolés. Cependant, le principe de compositionnalité, souvent attribué à Frege [Jan01] « Le sens d'une expression composée est une fonction du sens de ses parties. »,

défend l'idée que les mots s'influencent les uns les autres. Ainsi ne devraient-ils pas seulement être étudiés séparément mais aussi dans leur relation mutuelle. Pour ce faire, le projet Italodisco propose d'utiliser une combinaison de l'approche par les Tenseurs et de techniques d'apprentissage automatique, notamment les réseaux de neurones, dans un premier temps sur la composition de couples adjectif-nom en anglais. Par la suite, hors du cadre du stage, il pourra être envisagé de tester le modèle sur d'autres expressions composées (nom-nom ou verbe-objet par exemple) et dans d'autres langues.

Les Tenseurs sont des objets mathématiques de généralisation de matrices en trois dimensions ou plus, utilisés pour induire de la sémantique latente[1] à partir de co-occurrences multiples. Cette sémantique peut ensuite être utilisée pour la modélisation d'expressions compositionnelles. L'Analyse Sémantique Latente [LT+98] repose sur le principe suivant : Pour chaque lemme, on compte son nombre d'occurrences, puis on pondère ce nombre par l'indice *td-idf*, (« *term frequency – inverse document frequency »)*. Ainsi, on obtient une matrice dont les lignes sont les lemmes et les colonnes correspondent à la fréquence dans le document. Dans notre cas, le but est de créer un vecteur contenant toutes les valeurs de fréquences des termes apparaissant comme contexte des parties de nos couples adjectif-nom. Le vecteur est donc composé de nombres représentant les mots « contexte » d 'un constituant (un nombre unique est associé à chaque mot selon sa fréquence). Il capture sa signification intrinsèque. On crée ensuite une matrice contenant les vecteurs des deux termes. Cette matrice capture comment le contexte change la signification des constituants étudiés. Durant ce travail, en revanche, l'Analyse Sémantique Latente n'a pas été utilisée puisque la construction des matrices a été effectuée avec l'outil *Word2vec* décrit plus loin dans le mémoire. Cet outil fonctionne avec un algorithme fondé sur le même principe mais un peu différent[2].

Les réseaux de neurones donnent des résultats correspondant à l'état de l'art dans une variété de tâches du traitement du langage naturel. Ils permettent une approche fondée sur l'usage et non-supervisée. La performance du modèle que nous utilisons a été testée à l'aide d'une tâche de similarité existant pour les expressions de composition [M&L10].

Mes tâches

1/ La page officielle de l'indexation de la Sémantique Latente est trouvable à l'adresse suivante : http://lsa.colorado.edu/
2/ La bibliothèque est détaillée à l'adresse suivante : https://code.google.com/p/word2vec/

Le premier mois de mon stage fut consacré à la formation au python, à l'apprentissage automatique, à l'étude des matrices et vecteurs et de la régression logistique ; notions que, ayant un parcours principalement littéraire, je n'avais pas eu l'occasion d'étudier avant mon arrivée à l'IRIT. Par la suite, mon rôle fut d'adapter un modèle de réseau de neurones existant à nos données, de tester un ensemble de paramètres pour faire baisser le résultat d'une fonction de coût, d'entraîner le modèle afin d'extraire les meilleurs paramètres d'apprentissage (poids et biais) possibles et enfin d'en faire l'évaluation. Afin d'évaluer les résultats, un corpus d'annotations de similarité m'était fourni en format texte. Il m'a fallut le transformer pour le faire utiliser par l'auto-encodeur, calculer nos propres similarités sur ce même corpus puis comparer les résultats à l'aide de la corrélation de Spearman. A chaque phase d'entraînement, le modèle générait des paramètres de poids et de biais aléatoires. Ceux dont découlaient la meilleure corrélation de Spearman étaient finalement sélectionnés comme meilleurs paramètres. Pour valider notre évaluation, nous avons utilisé une autre méthode. Nous avons séparé le corpus en deux parties : 80% servait à l'entraînement et 20% au test. L'auto-encodeur devait calculer, à l'aide des meilleurs paramètres sélectionnés précédemment, un vecteur associé à un couple adjectif-nom à partir du vecteur de chacun de ses constituants. Le résultat de ce calcul a ensuite été comparé aux vecteurs adjectif-nom existants. Le *Mean Reciprocal Rank* a été utilisé afin de déterminer la performance de ce calcul.

I. Contexte

1. État de l'art

Sémantique formelle et sémantique lexicale

En sémantique formelle, un engouement renouvelé pour la sémantique lexicale s'est fait sentir récemment avec des travaux comme le Lexique génératif de Pustejovsky [Puj91], étudiant la nature distributive de la compositionnalité du langage naturel ou encore Asher [Ash11] et sa théorie des types. La limite de ces travaux est qu'ils ne sont pas généralisables et applicables à grande échelle. Notre approche combinant sémantique distributionnelle et approche par l'usage, c'est-à-dire ascendante (*bottom-up*) doit permettre à terme de fournir des informations sur de grandes quantités de données, à partir de celles-ci.

Sémantique distributionnelle

L'hypothèse distributionnelle de Harris [Har54] selon laquelle les mots qui apparaissent dans le même contexte sont sémantiquement semblables a donné lieu à de nombreux travaux dans le domaine de l'extraction sémantique automatique. Par exemple celui de Turney & Pantel [T&P10], développe des algorithmes pour tenter de capturer la sémantique des mots grâce à leur distribution.

Compositionnalité

L’étude de la compositionnalité découle naturellement de ces travaux. Un des premiers articles en la matière est celui de Mitchell & Lapata [M&L08]. Il compare différentes méthodes basées sur les vecteurs, principalement les modèles additifs et multiplicatifs. Ces modèles sont évalués sur une tâche de notation de la similarité entre des groupes verbes-noms. Nous développerons cet article plus tard car il a servi de base de comparaison à notre modèle.

Le modèle de Baroni & Zamparelli [B&Z10] étudie quant à lui les couples adjectif-nom

avec l'hypothèse que l'adjectif est une fonction linéaire sur le vecteur du nom qui mène au vecteur adjectif-nom. La transformation linéaire est ainsi une matrice extraite automatiquement d'un corpus à l'aide d'une méthode partielle des moindres carrés qui permet une généralisation et une pondération des données erronées.

Coecke & al [CSC10] prennent le parti d'étudier la phrase entière comme un vecteur issu du produit de Kronecker sur les vecteurs de tous les mots la composant. Cette méthode a ensuite été testée par Grefenstette & Sadrzadeh [GSa11] et [GSb11] avec l'idée que les adjectifs et les verbes, éléments multi-dimensionnels, agissent comme les filtres de leurs arguments.

Sens des mots dans leur contexte

Étroitement liés à la compositionnalité, il existe aussi plusieurs travaux sur la sémantique des mots en contexte. Erk & Padó [EPa08] et [EPb09] par exemple ont travaillé sur les « *selectional preferences* », c'est-à-dire la tendance d'un mot à apparaître en cooccurrence avec des mots de même champ lexical (exemple : l'adjectif *délicieux* apparaît très souvent avec des noms du champ lexical de la nourriture). Leur modèle multiplie le vecteur du mot avec un vecteur qui capture l'inverse des *selectional preferences* de l'argument. Ce modèle a inspiré les travaux de Thater et al. [TDP09] et [TFP10] qui l'ont étendu aux cooccurrences de second ordre. Enfin, Dinu & Lapata [D&L10] ont développé une méthode où le sens des mots est une distribution de probabilités sur les facteurs latents. Le sens en contexte est donc modélisé comme un changement par rapport à la distribution originale du sens.

Pour résumer, les travaux précédents se sont principalement concentrés sur la compositionnalité modérée d'un verbe sur son complément ou d'un couple adjectif-nom avec des méthodes plus ou moins complexes, mais la plus efficace semble rester la simple multiplication des vecteurs adjectif et nom.

Factorisation de tenseur

La modélisation de la langue naturelle par factorisation de tenseur a été plutôt fructueuse jusqu'à présent. Griesbrecht [Gie10] a notamment décrit une méthode de factorisation de tenseur pour construire un modèle distributionnel sensible à l'ordre des mots. Dans les travaux précédents de mon tuteur de stage, Van De Cruys [VDC10] et [VPK13], un modèle de factorisation de tenseur a été utilisé pour l'induction multi-dimensionnelle des *selectional preferences* et la modélisation de la compositionnalité. Ce modèle n'était néanmoins pas encore assez généralisant, d'où la poursuite des recherches durant ce stage.

Réseaux de Neurones

Les réseaux de neurones sont de plus en plus populaires dans le domaine du TAL. Bengio & al [BD+03], Mnih & Hinton [M&H07] et Collobert & Weston [C&W08] les ont utilisés pour la modélisation du langage naturel avec succès. Ils dépassent les difficultés que rencontre la représentation en n-grammes lorsqu'elle est confrontée à des données creuses (*sparseness*). Plusieurs chercheurs ont utilisé les réseaux de neurones pour tenter de modéliser la compositionnalité comme Socher & al [SH+12] et Tsubaki & al [TD+13].

Le modèle que le projet Italodisco étudie est un réseau de neurones tenseur (*Neural Tensor Network*) qui a notamment été utilisé pour la reconnaissance vocale par Yu & al [YD+13], l'extraction de connaissance par Socher & al [S+a13], et la détection des sentiments par Socher & al [S+b13]. Le modèle utilisé au cours du stage sera détaillé ci-après. Nous n'utilisons pas de tenseurs pour l'instant car il s'agit d'une évaluation préliminaire pour déterminer si le modèle est viable pour notre usage avant que les acteurs du projet Italodisco ne se risquent à l'utiliser avec des tenseurs.

2. Point de départ, article de Mitchell & Lapata

L'article de référence qui nous a servi de comparaison de résultats sur la tâche de similarité est celui de Mitchell & Lapata [M&L10] (« Composition in Distributional Models of Semantics »). Ce dernier nous a fourni un corpus de couples adjectif-nom pour tester notre modèle, une méthode d'évaluation et les résultats d'autres modèles sur cette même tâche afin de pouvoir nous comparer à d'autres.

Compositionnalité

Cet article fait une représentation mathématique des définitions de la compositionnalité en s'appuyant sur un état de l'art relativement fourni. La définition de la compositionnalité qu'il garde est celle de Partee [Par95], c'est à dire que le sens du tout est la fonction du sens de ses parties et de la façon dont ces parties se combinent syntaxiquement, à laquelle ils ajoutent la connaissance paradigmatique qu'apporte le locuteur lorsqu'il émet l'énoncé. Cela donne la représentation suivante où p est le sens du syntagme, R est la relation syntaxique entre les parties u et v , et K est le savoir complémentaire du locuteur : $p=f(u,v,R,K)$ Malgré tout, ils délaissent l'élément K afin de tester ce qui peut déjà être accompli sans la connaissance du contexte d'un point de vue fondamental. Ils avancent néanmoins la piste de l'utilisation de ressources telles que *WordNet* [Fel98] pour explorer cet aspect.

Modèles comparés

Le but de cet article est de comparer différents modèles avec la même méthode d'évaluation. Ils en choisissent donc neuf dont les quatre suivants que nous avons également utilisés à titre de comparaison avec notre modèle :

- Le modèle additif : $p=u+v$
- Le modèle additif pondéré qui considère par exemple que la tête d'un couple adjectif-nom est le nom et veut donc lui donner plus d'importance : $p=0{,}4\times u+0{,}6\times v$
- Le modèle multiplicatif : $p=u\times v$

- Le modèle basique qui ne considère que la tête du syntagme, un peu comme si on pondérait le modèle additif jusqu'à ce que *u* soit nul ($p = 0 \times u + 1 \times v$) : $p = v$

Ces modèles sont évalués sur une tâche de notation de similarité sémantique entre deux couples adjectif-nom par rapport à l'évaluation de cette similarité faite par des humains.

Corpus

Les syntagmes utilisés pour cette tâche de similarité, à savoir adjectif-nom, nom-nom, et verbe-objet, ont été obtenus à partir du British National Corpus (BNC) parmi les syntagmes les plus récurrents étiquetés par RASP (Briscoe & Carroll [B&C12]). Le défi était de ne pas faire des paires dont la similarité était trop facile à juger pour ne pas trop généraliser, mais pas non plus trop difficiles, de peur de n'avoir qu'une faible entente entre annotateurs et de rendre ainsi le test peu fiable.

Pour les paires très similaires, ont été prises seulement celles qui apparaissant au moins 100 fois dans le BNC (100 millions de mots). Ces paires devaient avoir également la caractéristique qu'en échangeant leurs têtes, la paire apparaissait toujours au moins 100 fois (ex: practical difficulty-economic problem >100 et practical problem-economic difficulty >100). L'hypothèse étant que cette recombinaison gagerait d'une redondance sémantique. Afin d'affiner les résultats, les auteurs de l'article ont utilisé une mesure de similarité basée sur un dictionnaire [Les86] pour classer les paires les plus similaires en fonction des renvois qu'elles se faisaient dans *WordNet*. Ils ont calculé la similarité entre les deux couples, et la somme de la similarité entre leurs constituants. Les 36 mieux notés étaient gardés dans la catégorie « hautement similaires », puis ces derniers ont été recombinés entre eux afin de donner les catégories « moyennement similaires » et « peu similaires ». Cette recombinaison avait pour but de discriminer les catégories selon la relation entre les mots et non un choix de vocabulaire différent pour ne pas perdre les annotateurs. On peut s'interroger sur le bien fondé de cette décision car les mots sont placés dans les catégories les uns par rapport aux autres, selon le nombre qu'ils sont dans chacune de ces catégories, et non jugés de façon absolue. Par exemple, les « moyennement similaires » peuvent être des « peu similaires » quand on compare à d'autres couples. Néanmoins, la position se défend et la recherche dans *WordNet* permet de pondérer ces possibles erreurs.

Similarité

Afin de comparer leur notes de similarité avec une notation humaine, une expérience a été menée sur 72 participants pour les couples adjectif-nom, 56 pour les couples nom-nom et 76 pour les couples verbe-objet, tous de langue maternelle anglaise et non-spécialistes de linguistique. Parmi ces participants, il y avait 73 femmes et 94 hommes de 17 à 66 ans (moyenne 31 ans). Chaque participant devait attribuer à la totalité des couples une note de similarité allant de 1 à 7. Afin de juger de leur entente, la technique de validation croisée par échantillonnage (*n-fold cross-validation*) [W&K91] a été utilisée ainsi que la corrélation de Spearman que nous détaillerons plus loin. Au final les interrogés ont trouvé la tâche très difficile mais ont été relativement constants dans leur notation. La corrélation de Spearman entre la notation calculée et celle des sujets d'étude a donné un résultat de 0,52 ; ce qui sert de point de comparaison aux auteurs de l'article. Le but est de s'approcher du résultat des sujets humains (0,52) et non de la notation exacte calculée. Le meilleur résultat est obtenu par le modèle multiplicatif.

Table 6
Correlation coefficients of model predictions with subject similarity ratings (Spearman's ρ) using a simple semantic space

Model	Adjective–Noun	Noun–Noun	Verb–Object
Additive	.36	.39	.30
Kintsch	.32	.22	.29
Multiplicative	.46	.49	.37
Tensor product	.41	.36	.33
Convolution	.09	.05	.10
Weighted additive	.44	.41	.34
Dilation	.44	.41	.38
Target unit	.43	.34	.29
Head only	.43	.17	.24
Humans	.52	.49	.55

Table 7
Correlation coefficients of model predictions with subject similarity ratings (Spearman's ρ) using the LDA topic model

Model	Adjective–Noun	Noun–Noun	Verb–Object
Additive	.37	.45	.40
Kintsch	.30	.28	.33
Multiplicative	.25	.45	.34
Tensor product	.39	.43	.33
Convolution	.15	.17	.12
Weighted additive	.38	.46	.40
Dilation	.38	.45	.41
Head only	.35	.27	.17
Humans	.52	.49	.55

Tableau 1: Résultats de la corrélation de Spearman pour les divers modèles testés par Mitchell et Lapata

3. Les réseaux de neurones

Généralités

Un réseau de neurones est un ensemble de fonctions optimisé par des techniques d'apprentissage automatique. On les utilise principalement dans le domaine des statistiques mais pas seulement. Ils fonctionnent sur le principe d'induction depuis les données à l'aide d'un entraînement. Le but initial était d'imiter le fonction du cerveau humain à travers ses neurones et ses synapses, en particulier les capacités d'apprentissage, de mémorisation et de traitement d'informations incomplètes. Chaque processeur du réseau calcule une sortie unique à partir des informations qu'il reçoit en entrée puis les sorties sont mises en commun. C'est une vision extrêmement simplifié du fonctionnement des neurones humains qui reçoivent des signaux de centaines de milliers de synapses à la fois contre une dizaine maximum pour un « neurone » artificiel et la communication entre eux est très complexe. D'abord à la mode, puis laissés de côté, les réseaux de neurones ont connus un renouveau en 1985 avec l'arrivée de la rétropropagation en gradient utilisée notamment dans le modèle de perceptron multicouche toujours utilisé aujourd'hui. Ils sont particulièrement appréciés pour leur capacité de classification et de généralisation. C'est cette dernière qui nous intéresse afin d'induire et de généraliser la compositionnalité entre les membres d'un couple adjectif-nom. Pour être efficace en revanche, un réseau de neurones doit disposer de grandes quantités de données pour ne pas faire de sur-apprentissage sur des données non-représentatives. Il existe plusieurs modèles de réseau de neurones. Celui que nous utilisons est basé sur un autoencodeur. Il s'agit d'un modèle de type « *denoising* » c'est-à-dire « débruitant » en français. Nous utiliserons le nom anglais faute d'une traduction officielle plus heureuse.

Denoising auto-encoder

Le principe d'un autoencodeur (Figure 1) est qu'il apprend grâce à une phase d'entraînement une représentation compressée et distribuée d'un jeu de données (ce qu'on appelle l'encodage) afin notamment d'en réduire la dimension. La forme la plus connue est le perceptron multicouche créé pour pouvoir résoudre des problèmes non linéaires. Il comprend une couche d'entrée à laquelle on applique une fonction pour obtenir une à plusieurs couches cachées auxquelles on applique une

fonction pour calculer une couche de sorties, se voulant être la reconstruction de la couche d'entrée.

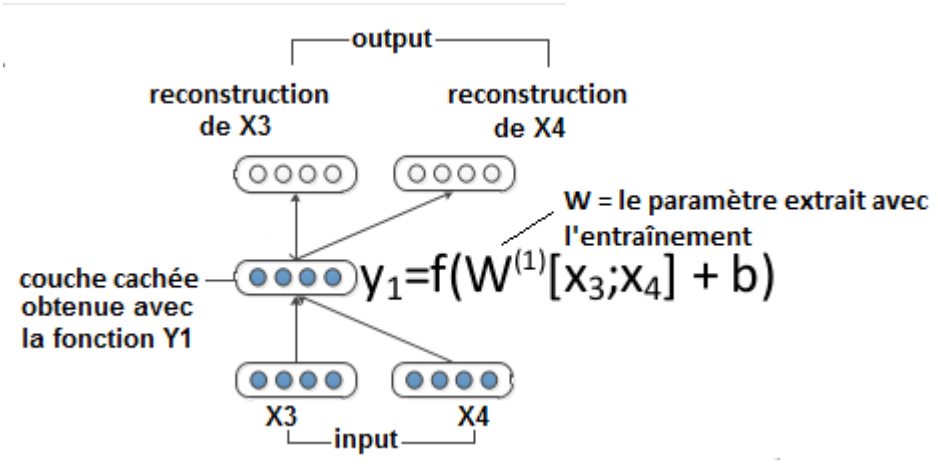

Figure 1: Un exemple de modèle de réseau de neurones : l'autoencodeur

Le denoising autoencodeur est un modèle non-supervisé conçu par Vincent et al.[3] [VL+08] pour rendre le modèle précédent robuste en cas de données bruitées. Sans ce système, le modèle pourrait sur-apprendre et reproduire exactement les données d'entrée en sortie, qu'elles soient bruitées ou non, ce qui serait mauvais pour la généralisation. Pour remédier à cela, les données d'entrée sont bruitées, c'est-à-dire aléatoirement rendues nulles (égales à zéro) selon une probabilité p ou à l'aide d'une Gaussienne. La couche reconstruite est alors calculée à partir des données bruitées. On calcule également une fonction de coût en comparant la couche reconstruite aux données non bruitées initiales afin d'estimer la perte. La fonction de coût doit donc être la plus basse possible. Voir Figure 2 et 3 ci-après.

3. Le modèle en question peut être trouvé à l'adresse suivante : http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf

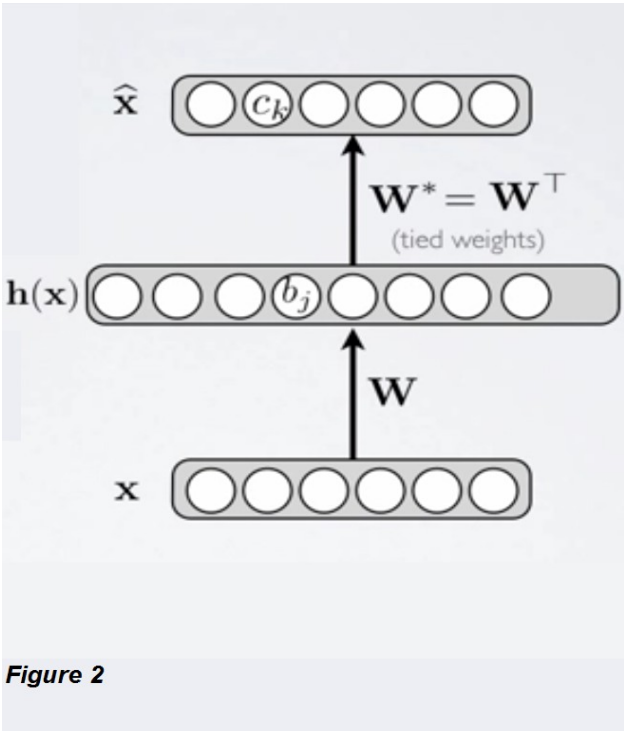

Figure 2: Autoencodeur classique

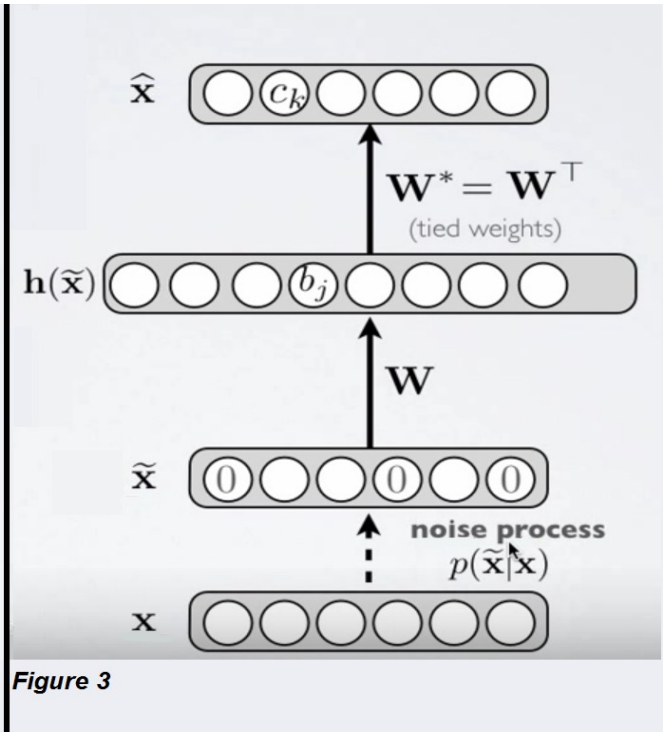

Figure 3: Denoising Autoencodeur. La couche bruitée permet de prendre obligatoirement en compte le bruit possible des données.

Notre modèle reçoit en entrée les vecteurs A et N associés respectivement à des adjectifs et à des noms et s'entraîne sur les vecteurs AN des couples adjectif-nom créés à partir du corpus. Par conséquent, les vecteurs de contexte sémantiques réduits (ou compressés) que le modèle calcule pour la combinaison adjectif-nom sont construits à partir des données du corpus.

II. Notre système de réseau de neurones

1. Outils et librairies

Python 2.7, Anaconda, pickle, SciPy and NumPy

Notre modèle de réseau de neurones a été développé en python 2.7, un langage très utilisé lorsqu'il s'agit de créer des prototypes grâce à la vitesse d'écriture du code et à sa lisibilité. Il existe en outre une très grande variété de bibliothèques qui permettent de tester rapidement si une solution est possible puisqu'il suffit d'importer les fonctions appropriées, comme nous l'avons fait avec le package Anaconda. Voici une présentation des principales bibliothèques utilisées au cours de ce stage.

Les données que nous utilisons sont très lourdes, nous avons donc utilisé le format Pickle qui transforme un objet python en fichier binaire dans un processus de sérialisation. Ces fichiers peuvent alors être reconstruits plus tard. Pour accélérer encore le traitement de ces données, nous avons utilisé Cpickle qui implémente le même algorithme mais en langage C au lieu de Python, ce qui le rend beaucoup plus rapide.

Utiliser des matrices nous a également poussés à utiliser SciPy et NumPy. Ces derniers permettent de manipuler aisément des tableaux multidimensionnels, notamment grâce à la fonction « array » de NumPy, et de faire de nombreux calculs pré-codés tels que les manipulations de matrices ou encore la corrélation de Spearman.

Theano

Theano est une bibliothèque Python qui permet de définir, d'optimiser et d'évaluer des expressions mathématiques impliquant des tableaux multi-dimensionnels. Ainsi, la communauté a

déjà modélisé de nombreuses fonctions , dont la régression logistique utilisée dans le modèle de réseau de neurones que nous désirons utiliser. De plus, un exemple de « *denoising autoencoder* » est entièrement disponible gratuitement sur le site de la documentation de Theano[4].

Ce modèle prend donc les données d'entrée ***x***, et les encode dans la couche cachée ***y*** par une fonction non-linéaire telle que la sigmoïde (***s***). On pondère à l'aide de paramètres générés aléatoirement qui feront partie des paramètres « appris » (***W***) avec le deuxième paramètre appelé biais (***b***) puis leurs doubles ***W'*** et ***b'*** utilisés lorsqu'on reconstruit la couche de sortie.

$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Ensuite, il reconstruit les données dans une couche de sortie ***z*** en appliquant la formule inverse.

$$\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$$

Le taux d'erreur ou « coût » est calculé avec une formule qui permet de considérer les données d'entrée comme des vecteurs d'objets ou comme des vecteurs de probabilité d'objets :

$$L_H(\mathbf{x}, \mathbf{z}) = -\sum_{k=1}^{d} [\mathbf{x}_k \log \mathbf{z}_k + (1 - \mathbf{x}_k) \log(1 - \mathbf{z}_k)]$$

Ce « coût » doit être le plus bas possible mais il ne sera jamais nul car la représentation compressée fait toujours perdre des informations. Néanmoins, le but est d'apprendre une généralisation et non de pouvoir reproduire exactement les données d'entraînement. On peut jouer sur différents paramètres pour faire baisser cette fonction de coût et les essais à ce sujet seront détaillés dans la partie II.3 .

4.http://deeplearning.net/tutorial/dA.html#daa

2. Corpus d'entraînement et de test

Corpus

Notre corpus est extrait de Wikipédia. Outre sa particularité d'être très imposant au niveau de la quantité de données, il n'est pratiquement composé que de texte au mode affirmatif, des informations brutes, ce qui nous épargne beaucoup de mauvaises interprétations sémantiques dues à la modalité. De plus, le contenu est entièrement libre et gratuit, ce qui en fait un outil très populaire pour la recherche en traitement automatique des langues. Ce corpus a ensuite été lemmatisé et étiqueté grâce au *Stanford NLP pipeline*[5], un outil programmé en java qui prend en entrée un texte brut, le lemmatise et l'étiquette, même les noms propres, grâce au NER (*name entity recognizer*), aussi connu sous le nom de *CRF Classifier,* qui analyse les chaînes de champs inconnus. Il normalise également les dates et heures et s'utilise avec très peu de lignes de codes. Il possède différentes options d'analyse qu'on peut aisément désactiver ou activer comme l'analyse de sentiment. Cet outil utilise pour l'anglais la liste d'étiquettes appelée « *Penn Treebank tag set* »[6] [MMS93] et [San90] composée de 36 étiquettes.

Dans le corpus, les mots ont ainsi pu être transformés en indices selon leur fréquence et les adjectifs et les noms ont pu être sélectionnés. L'outil w*ord2vec* a ensuite été appliqué afin d'obtenir automatiquement les vecteurs associés à chaque adjectif, nom et couple adjectif-nom.

Word2vec

Word2vec est un outil dont l'algorithme a été écrit par Tomas Mikolov. Nous utilisons l'implémentation de la bibliothèque Gensim[7] créé par Radim Rehurek. Cet outil est fondé sur les sacs-de-mots continus. Cette représentation de texte est basée sur le principe qu'un document est représenté par l'histogramme des occurrences des mots le composant. À chaque mot, on associe sa fréquence dans le document, créant ainsi un vecteur de la même taille que le dictionnaire du

5. Pour plus d'information concernant l'outil, consulter ce site :
http://nlp.stanford.edu/software/corenlp.shtml

6. La liste entière de ces étiquettes est disponible ici :
https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

7. La bibliothèque peut être trouvée à cette adresse:
https://radimrehurek.com/gensim/models/word2vec.html

document qu'il conviendra de normaliser. *Word2vec* utilise également les architectures *Skip-Grams* (voir formule 1) [GA+06] qui fonctionnent comme les n-grammes mais où les tokens adjacents peuvent être « sautés ». Par exemple pour les 2-skips-bigrammes sur la phrase « Insurgents killed in ongoing fighting », on obtiendra :

Bi-grams={insurgents killed, killed in, in ongoing, ongoing fighting}

2-skip-bi-grams={insurgents killed, insurgents in, insurgents ongoing, killed in, killed ongoing, killed fighting, in ongoing, in fighting, ongoing fighting}

Donc, si on cherche à récupérer les couples adjectif-nom de la phrase « She has gorgeous curly red hair. », on obtiendra les couples « red hair », mais aussi « curly hair » et même « gorgeous hair ».

$$\{w_{i_1}, w_{i_2}, \ldots w_{i_n} \mid \sum_{j=1}^{n} i_j - i_{j-1} < k\}$$

Formule 1: Calcul des k-skip-n-grams pour une phrase dont les mots vont de W1 à Wm

On utilise *Word2vec* pour convertir les mots en vecteurs directement ou à partir d'un fichier existant. On peut ainsi lui faire exécuter des tâches syntaxiques ou sémantiques comme le calcul de la similarité entre deux mots, par exemple :

```
>>> model.similarity('woman', 'man')
0.73723527
```

ou la détection de l'élément d'une liste le plus similaire à ce qu'on cherche avec des conditions, par exemple :

```
>>> model.most_similar(positive=['woman', 'king'], negative=['man'])
[('queen', 0.50882536), …]
```

Le modèle peut être entraîné sur un corpus donné afin d'obtenir des résultats plus spécifiques. Dans notre cas, il a été utilisé pour transformer les mots en indices selon leur fréquence, construire les vecteurs des adjectifs et des noms ainsi que les vecteurs des couples adjectif-nom.

Nous aurions pu utiliser des vecteurs qui encodent le contexte original de chaque mot sélectionné (les mots qui l'entourent dans une fenêtre) mais cela aurait généré des vecteurs creux et trop volumineux (environ 10 000 valeurs) alors que *Word2vec* réduit la densité des vecteurs à 100 valeurs, ce qui les rend possibles à traiter par l'algorithme de notre réseau de neurones. Le modèle de base (*baseline*) qui tient compte seulement la tête de chaque couple (le nom), utilise le vecteur brut obtenu par *Word2vec* et son résultat tout à fait honnête nous conforte dans l'idée que cet outil capture efficacement la sémantique des mots.

3. Modèles et paramètres testés

La première étape de nos expérimentations consistait à faire baisser le résultat de la fonction de coût. Pour cela, nous avons testé différentes valeurs de *learning rate*, à savoir : 1 ; 0,1 ; 0,01 ; 0,02 et 0,001. On calcule également l'efficacité du modèle avec ces différentes valeurs de *learning rate* grâce à la corrélation de Spearman. Les valeurs ci-dessus ont été testées avec différentes versions de l'autoencodeur correspondant à deux fonctions non-linéaires différentes permettant de généraliser les résultats.

La première utilisait une fonction d'unité linéaire rectifiée (*rectified linear unit*), une méthode de réduction du tenseur par création d'une matrice composée de zéros. On sélectionne le maximum, élément par élément, entre cette matrice de zéros et le résultat du calcul du réseau de neurones afin d'éliminer toutes les valeurs négatives. Avec cette méthode, les résultats de la fonction de coût obtenus variaient de 0,86 à 0,36 et la corrélation de Spearman entre une valeur infinitésimale et 0,25 suivant la valeur de *learning rate* utilisée.

La deuxième méthode utilisait la fonction sigmoïde, une fonction de régression logistique. La régression logistique permet de prédire une variable à partir de données. Cette méthode donne des valeurs de prédictions entre 0 et 1. Avec ce modèle, on obtient des résultats de fonction de coût entre 0,86 et 0,23 et une corrélation de Spearman entre 0,09 et 0,44, le meilleur résultat étant obtenu avec le *learning rate* de 0,01. C'est donc la fonction de sigmoïde avec cette valeur de *learning rate* qui a été sélectionnée pour être également évaluée par le *Mean Reciprocal Rank*.

T.maximum		
learning rate	fonction de coût	Spearman
1	0,86	<0,01
0,1	0,44	<0,01
0,01	0,38	0,14
0,02	0,39	0,11
0,001	0,36	0,25

Tableau 2: Résultat le plus bas de la fonction de coût et le plus haut de la corrélation de Spearman selon le learning rate après entraînement de 50 itérations pour la fonction Tensor.maximum de Theano

Sigmoïde		
learning rate	fonction de coût	Spearman
1	0,86	<0,01
0,1	0,44	0,09
0,01	0,23	0,44
0,02	0,25	0,43
0,001	0,23	0,43

Tableau 3: Résultat le plus bas de la fonction de coût et le plus haut de la corrélation de Spearman selon le learning rate après entraînement de 50 itérations pour la fonction sigmoïde de Theano

III. Évaluation et résultats

Dans ce chapitre, nous présentons nos deux méthodes d'évaluation, leurs résultats et les conclusions que nous avons pu en tirer. La première méthode consiste à évaluer les modèles sur une tâche de notation de similarité entre deux couples adjectif-nom et à comparer cette notation à celle qu'ont donné des humains pour ces mêmes couples. La deuxième consiste à reconstruire le vecteur de contexte d'un couple adjectif-nom à partir des vecteurs de contexte de l'adjectif et du nom composant ce couple. Ce vecteur reconstruit est alors comparé aux vecteurs calculés par *Word2vec* pour voir si celui dont il est le plus proche est bien celui qu'il fallait trouver.

1. 1ère méthode de test : corrélation de Spearman

Nous avons évalué notre modèle sur deux tâches différentes. La première, comme dans l'article de Mitchell & Lapata [M&L10] est une tâche de calcul de similarité. Nous avons récupéré le corpus de notation de l'article et en avons extrait celles des duos de couples adjectif-nom uniquement. Ensuite, en recherchant les vecteurs de chaque adjectif et de chaque nom des duos, par exemple pour le duo [Adj1 Nom1; Adj2 Nom2], notre modèle a calculé les vecteurs adjectif-nom de chaque couple (vAN1 et vAN2), les a normalisé, puis a calculé le cosinus de similarité entre les deux vecteurs. Ce cosinus est ensuite utilisé comme notation de similarité du duo. La corrélation de Spearman a ensuite été calculée entre la liste de nos notations et celle des notateurs humains du corpus afin d'en estimer la proximité malgré le fait que nos notes aillent de -1 à 1 et les leurs de 1 à 7. Cette méthode permet justement de comparer les notations malgré les échelles différentes. Nous avons ensuite calculé les notes de similarités avec les modèles de l'article cités précédemment, à savoir additif, additif pondéré, multiplicatif et le modèle basique puis comparé leur notation à celle des humains avec cette même corrélation. Ainsi, nous avons pu mesurer notre autoencodeur aux modèles existants avec des résultats comparables car obtenus à partir du même corpus.

La corrélation de Spearman (nommée d'après Charles Spearman) est une méthode statistique visant à déterminer si des fonctions sont corrélés. Il existe également la corrélation de Pearson qui s'applique aux relations linéaires. La corrélation de Spearman, elle, s'applique aux relations non-linéaires. Elle est également appelée « corrélation de rang » puisqu'elle observe la différence des rangs entre les individus des deux éléments comparés. On peut ainsi détecter des relations dites « monotones » (croissantes ou décroissantes) qu'elle que soit leurs formes (linéaire, exponentiel,

puissance). Le résultat de cette corrélation oscille entre -1 et +1 avec :

1 = corrélation totale

0 = aucune corrélation

-1 = corrélation inversée

$$\rho(X,Y) = 1 - \frac{6 \cdot \sum_{i=1}^{N} \left[r(X_i) - r(Y_i) \right]^2}{N^3 - N}$$

avec

$r(X_i)$: rang de X_i dans la distribution $X_1 \ldots X_N$

$r(Y_i)$: rang de Y_i dans la distribution $Y_1 \ldots Y_N$

Formule 2: Calcul de la corrélation de rang de Spearman entre deux éléments X et Y

La corrélation de Spearman doit être validée par un test de significativité car, selon le nombre de données observées, on pourrait ne disposer que de données non-représentatives. Ce coefficient ainsi que la corrélation elle-même est calculé automatiquement par la fonction *spearmanr* de la librairie SciPy. Malgré tout, notre corpus de test étant composé de plus de 1900 occurrences, les chances que nos données ne soient pas représentatives restent relativement minces. D'ailleurs, l'indice de données non représentatives est allant de 0 (données représentatives) à 1 (données non représentatives) est pour notre modèle encodeur p1 de 8.9946576975570899e-98.

A titre d'information, la corrélation entre les notateurs humains était de *0,52*. Elle a été calculée sur des notes de similarités entre couples adjectif-nom et représente l'accord inter-annotateur. Elle n'est donc pas tout à fait comparable mais elle donne un ordre d'idée. Les notateurs n'étant pas tous d'accord et leurs notes étant toutes présentes dans le corpus de similarité, la corrélation de Spearman ne pourra de toutes façons pas atteindre des valeurs exceptionnelles.

Nous avons gardé deux lots de paramètres (poids et biais) différents pour notre autoencodeur. En effet, nous avons utilisé une deuxième méthode d'évaluation (voir « 2ème méthode : Mean Reciprocal Rank ») et il apparaît que certains paramètres donnant une corrélation de Spearman d'au moins *0,45* soient mauvais dans cette deuxième tâche alors que certains paramètres donnant une corrélation de Spearman un peu plus basse mais toujours correcte donnent un bien meilleur résultat. Ce deuxième lot de poids et biais est désigné dans le tableau suivant par l'appellation « autoencodeur p2 » alors que celui donnant la meilleure corrélation de Spearman est appelé « autoencodeur p1 ».

modèles	corrélation de Spearman
basique	0,425
additif	0,449
additif pondéré	0,463
multiplicatif	0,271
autoencodeur p1	0,457
autoencodeur p2	0,444

Tableau 4: Résultats de la corrélation de Spearman entre les similarités calculées par les modèles testés et les notations de similarité des humains

Afin d'analyser sur quels types de duo de couples adjectif-nom notre modèle diffère de la notation humaine, nous avons créé les visualisations ci-dessous sur 1000 notes seulement pour des raisons de lisibilité. La première figure montre la corrélation de Spearman du meilleur système présenté par Mitchell & Lapata [M&L10], l'additif pondéré, et la deuxième, celle de notre autoencodeur p1.

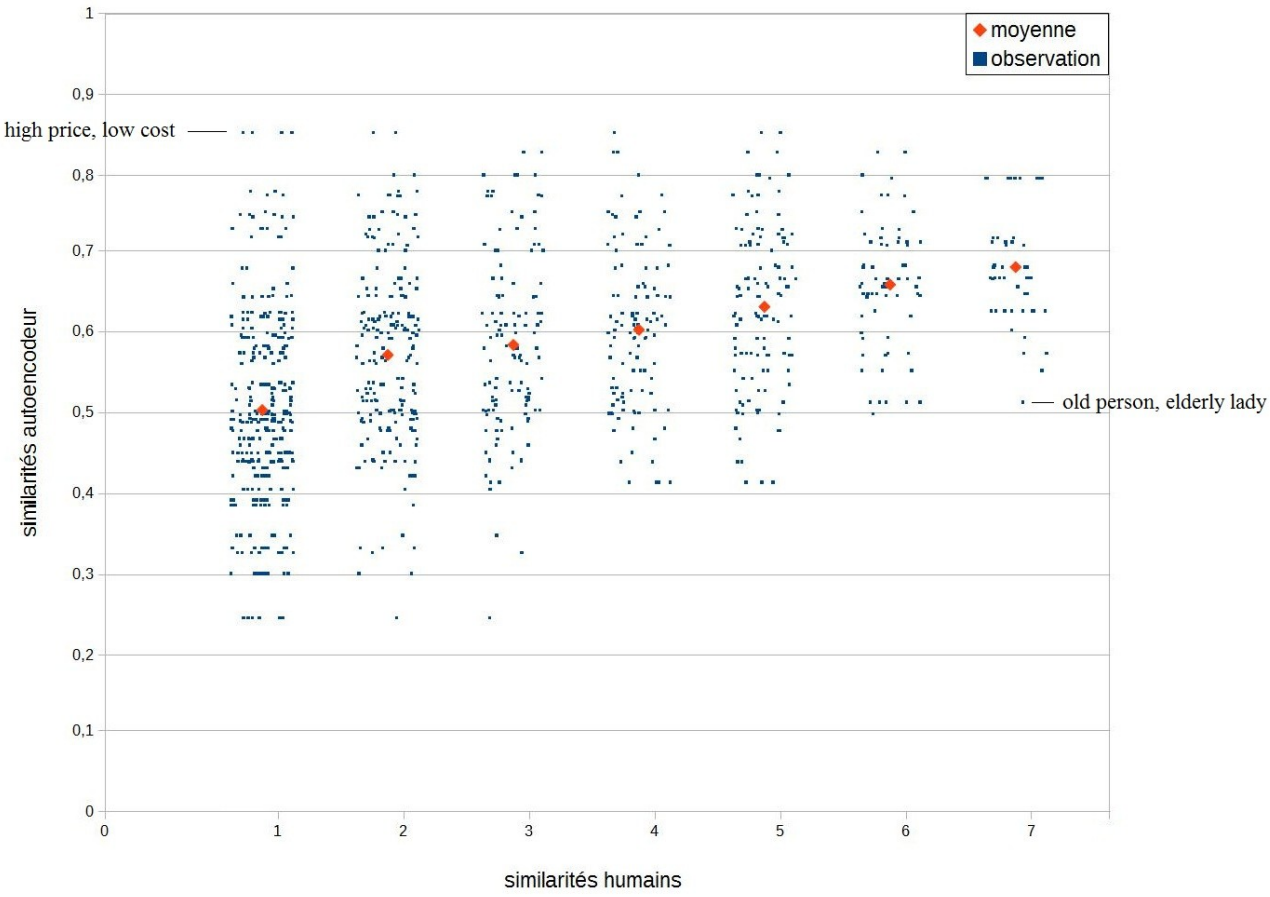

Figure 4: Visualisation de la corrélation de Spearman du modèle additif pondéré sur 1000 notes de similarité. Les points rouges représentent la moyenne des notes du modèle correspondant à une note humaine (de 1 à 7)

Dans le modèle additif pondéré, nous avons relevé un exemple de duos de couples adjectif-nom ayant un mauvais score de similarité selon les humains et haut selon le modèle et un pour le cas inverse. Il s'agit respectivement du duo [« high price », « low cost »] et du duo [« old person », « elderly lady »]. Le modèle analyse le contexte. Ainsi donc, on peut supposer que « high price » et « low cost » sont utilisés dans des contextes tellement similaires que le modèle ne voit pas qu'il s'agit d'antonymes. Ils sont en revanche notés comme « pas du tout similaires » par les humains.

Inversement, « elderly lady » a beau être le méronyme de « old person », ce qui vaut au duo une bonne note selon les humains, les deux couples doivent apparaître dans des contextes assez différents. Le manque de connaissance pragmatique empêche alors le modèle de repérer la proximité sémantique du duo.

Une comparaison avec les résultats de notre modèle permet d'encourager les réflexions sur la difficulté du modèle vectoriel à repérer le duo antonyme [« high price », « low cost »] et à mettre en lumière les quelques échecs liés au manque de connaissance pragmatique. Malgré tout, nous pouvons constater que les notes de similarité des modèles suivent relativement bien celles des humains.

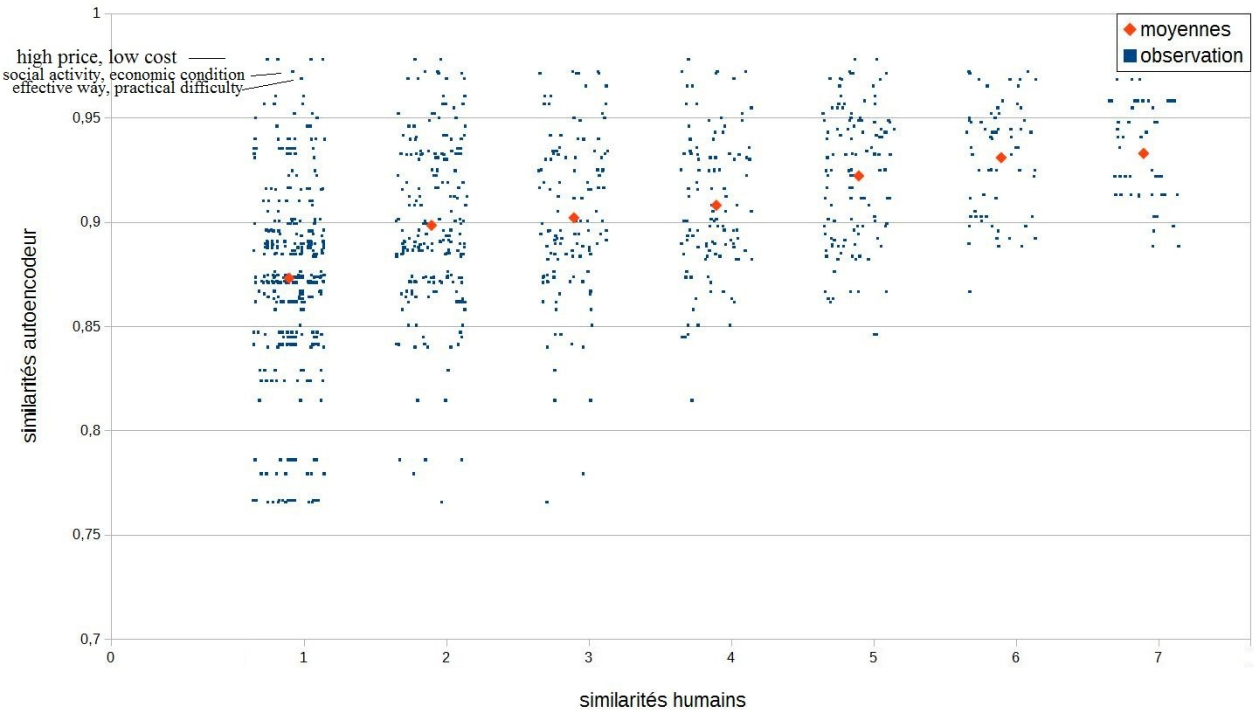

Figure 5: Visualisation de la corrélation de Spearman de l'autoencodeur p1 sur 1000 notes de similarité. Les points rouges représentent la moyenne des notes du modèle correspondant à une note humaine (de 1 à 7)

Comme pour le modèle additif nous avons relevé pour l'autoencodeur les duos dont la note différait le plus entre humains et modèle. Pour commencer, parmi les duos bien notés par le modèle mais pas par les humains, nous retrouvons le duo d'antonymes [« high price », « low cost »]. Nous trouvons également [« social activity », « economic condition »] ou [«effective way », « practical difficulty »] qui n'ont pas véritablement de lien sémantique au premier abord mais qui doit probablement apparaître dans un contexte similaire. Pour ce modèle, la différence inverse, c'est-à-dire les cas où le duo obtient une bonne note par les humains et mauvaises par l'autoencodeur, est beaucoup moins flagrante. Il serait néanmoins intéressant de faire une étude approfondie pour savoir si les modèles vectoriels peinent à repérer les antonymes en général ou si ce n'est le cas que pour ce duo.

Pour l'anecdote, dans le modèle autoencodeur p2, nous avons noté un duo étonnant bien noté par les humains mais mal noté par le modèle : [« major issue », « american country »]. Cet exemple incongru nous rappelle la subjectivité des notateurs humains et l'importance des connaissances pragmatiques qu'il manque à nos modèles.

2. 2ème méthode: Mean Reciprocal Rank

Afin de nous donner une autre estimation de nos performances, nous avons utilisé le calcul du *Mean Reciprocal Rank* (MRR). Cette méthode de statistiques est très utilisée pour calculer la pertinence des résultats de moteurs de recherche. Elle consiste à récupérer, dans une liste de résultat ordonnée, le rang *x* du résultat attendu puis d'en prendre l'inverse (*1/x*). Cette opération est effectuée sur plusieurs requêtes puis la moyenne de tous les rangs inversés obtenue est calculée.

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}.$$

Pour appliquer cette méthode, le corpus d'entraînement a été divisé aléatoirement en deux parties de 80 % et 20 % s'excluant l'une l'autre pour constituer respectivement un nouveau corpus d'entraînement et un nouveau corpus de test. La méthode *k-fold* n'a pas été utilisée ici car le lot de poids et biais obtenu avec l'entraînement servait aussi pour le calcul de la corrélation de Spearman, or il fallait utiliser le même corpus pour les deux évaluations et les deux corpus (entraînement et test) ne devaient jamais se recouper.

Le corpus de test comportait un fichier contenant la matrice des vecteurs de contexte des occurrences de couples adjectif-nom (*Van*) et un fichier contenant les indices des adjectifs et des noms (*a* et *n*) pour chaque couple adjectif-nom. Après une phase d'entraînement sur les 80 %, pour chaque couple *AN* contenu dans la matrice de vecteurs du corpus de test (20%), nous avons récupéré les indices des adjectifs et des noms (*a* et *n*) ainsi que leurs vecteurs individuels (*Va* et *Vn*). À partir de ces derniers, nous avons fait calculer à notre modèle *Van'*, le vecteur combiné de chaque couple adjectif-nom normalisé. Ce qui nous donnait donc pour chaque vecteur *Van* (inconnu de l'autoencodeur), un vecteur calculé *Van'*. Ensuite, chaque vecteur *Van'* calculé a été comparé à tous les vecteurs *Van* de la matrice de vecteur à l'aide de la similarité cosinus (*cos(Van'1, Van1)*, puis *cos(Van'1, Van2)* etc). La liste des cosinus pour chaque couple adjectif-nom a subséquemment été triée par ordre décroissant afin d'observer de quel vecteur *Van* le vecteur *Van'* correspondant était le plus proche. Pour finir, nous avons parcouru la liste des cosinus pour obtenir le rang du vecteur *Van* à trouver dans la liste. Ce rang était finalement récupéré puis inversé (*1/rang*) puis ajouté aux autres rangs inversés de chacun des autres vecteurs *Van'* pour calculer la moyenne : le MRR. Pour résumer, il s'agit de reconstruire des vecteurs de couple adjectif-nom à partir du vecteur attribué à l'adjectif et du vecteur attribué au nom et d'en mesurer la pertinence en les comparant à l'ensemble des vecteurs

Van originaux. Notre premier résultat portant sur l'autoencodeur p1 était peu convainquant.

Nous avons donc calculé le MRR sur d'autres lots de poids et biais retenus et avons remarqué sans trop savoir s'il s'agit d'une constante que certains lots de poids et biais donnant une corrélation de Spearman supérieure à *0,45* donnaient un MRR très faible alors que certains donnant une corrélation de *0,44* étaient bien meilleurs sur cette deuxième tâche. Nous en avons donc retenu un pour l'exemple, référencé dans le tableau ci-dessous sous l'appellation autoencodeur p2. Nous avons ensuite procédé au calcul du MRR pour les autres modèles présentés par l'article de Mitchell & Lapata [M&L10]. Nous pouvons constater que l'autoencodeur p2, même s'il est battu de loin par les modèles additif et additif pondéré, reste meilleur que le modèle basique qui n'inclut pas de compositionnalité puisqu'il ne prend en compte que le nom. Il est donc possible que notre prise en compte de la compositionnalité soit tout de même une amélioration dans cette tâche.

modèles	Mean Reciprocal Rank
basique	0,123
additif	0,397
additif pondéré	0,327
multiplicatif	0,002
autoencodeur p1	0,081
autoencodeur p2	0,188

Tableau 5: Résultats du calcul du MRR pour tous les modèles testés sur 30888 vecteurs.

Afin de mieux se représenter la répartition des bonnes réponses de l'autoencodeur p2, nous avons répertorié dans un graphique la proportion des vecteurs arrivant dans les 10 premiers rangs ainsi que celle de ceux arrivant à un rang supérieur à 100 sur les 30888 rangs possibles. Pour l'autoencodeur p2 au MRR de 0.188, on trouve donc près de 3868 vecteurs parfaitement calculés (rang 1), soit environ 13 %, et au total 17290 vecteurs sont classés à un rang inférieur ou égal à 10 sur 30888 (soit 56% environ). En revanche, 14527 vecteurs ont un rang supérieur à 100. À titre informatif, ce nombre atteint les 21046 avec l'autoencodeur p1.

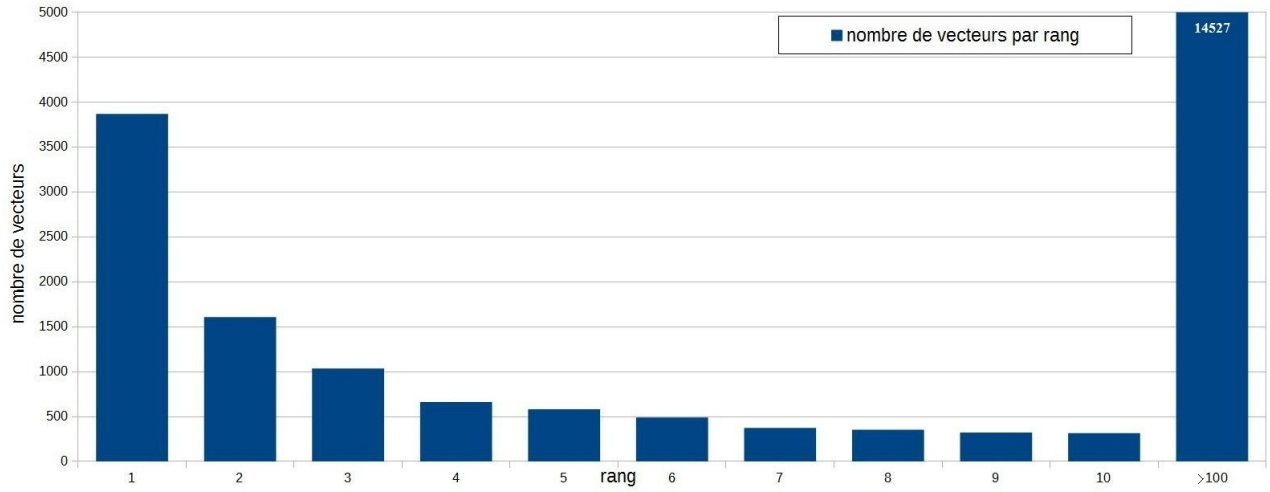

Figure 6: Répartition des rangs du nombre de vecteurs atteignant les 10 premiers rangs et de ceux dépassant le rang numéro 100 pour l'autoencodeur p2.

3. Analyse

La tâche du MRR est complexe et exigeante puisqu'il s'agit de reconstituer parfaitement des vecteurs. Or notre modèle perd des informations par la compression, comme le montre le résultat de la fonction de coût. Il est donc logique que cette tâche donne des résultats moins bons que pour une tâche plus généralisante qu'est la notation de la similarité, utilisée pour le calcul de la corrélation de Spearman.

Pour cette dernière, nos résultats ne jurent pas avec les modèles précédents, ni avec l'accord qu'ont les notateurs entre eux. Notre modèle n'est battu que de peu par les modèles additif et additif pondéré. Nous pouvons néanmoins remarquer que le modèle multiplicatif qui semblait, dans l'article de Mitchell & Lapata [M&L10], être le plus performant est à la traîne dans nos tests. Dans l'article en question et contrairement à nos travaux, les vecteurs étaient construits avec une autre méthode que notre modèle de sémantique latente, les skip-grammes. Il se trouve que cette méthode conserve dans les vecteurs des valeurs négatives. Or, avec la multiplication, les valeurs changent de signe, ce qui ne permet pas de retenir efficacement la sémantique. Cela explique donc la grande différence de résultats de ce modèle lorsqu'il utilise notre corpus de vecteurs avec celui de Mitchell & Lapata [M&L10].

Conclusion

Bilan

Dans ce mémoire, nous nous sommes intéressés au test d'un modèle de réseau de neurones pour le traitement de la compositionnalité.

Nous avons premièrement établi un état de l'art des travaux existant en Traitement Automatique des Langues pour cet aspect de la sémantique. Nous avons ensuite présenté les modèles de Mitchell & Lapata [M&L10] (additif, additif pondéré, multiplicatif et basique) que nous allions tester sur le même corpus que nous afin de comparer nos résultats aux leurs. Nous avons évoqué l'adaptation de ce corpus car nous n'en avons gardé que la partie concernant les couples adjectif-nom et nous avons supprimé les doublons. Enfin, nous avons exposé le modèle de réseau de neurones que nous avons utilisé : l'autoencodeur.

Dans la deuxième partie, nous avons expliqué que notre corpus venait de Wikipédia et qu'il avait subit un étiquetage morphosyntaxique. Puis l'outil *Word2vec* nous a permis de constituer automatiquement les vecteurs de couple adjectif-nom utilisés pour l'entraînement puis pour le test du *Mean Reciprocal Rank*. Nous avons également parlé des l'entraînement de l'autoencodeur pour sélectionner le meilleur learning rate ainsi que pour obtenir les meilleurs poids et biais.

Enfin, la troisième partie exposait les deux tâches sur lesquelles le modèle était testé. La première consistait en l'attribution de notes de similarité pour des duos de couples adjectif-nom (par exemple : [« practical difficulty », « economic problem »]). Ces notes étaient ensuite, avec la corrélation de Spearman, comparées aux notes données par un panel d'humains aux même duos de couples adjectif-nom. Les résultats de ce test étaient plutôt concluant mais pas révolutionnaires. Pour la deuxième, il s'agissait de reconstruire des vecteurs de couple adjectif-nom à partir du vecteur attribué à l'adjectif et du vecteur attribué au nom. Le résultat de cette tâche était évalué à l'aide d'une mesure de moyenne, le *Mean Reciprocal Rank.* Le bilan de ce dernier test est mitigé. Notre modèle est battu, et de loin, par les modèles additif et additif pondéré mais reste meilleur que les autres, notamment le modèle basique que nous devions absolument dépasser pour montrer que la prise en compte de la compositionnalité apportait quelque chose.

Perspectives

Différents tests peuvent encore être menés sur le modèle actuel. Pour commencer, dans ce stage, nous aurions dû essayer de remplacer la fonction non linéaire sigmoïde par la tangente mais le temps nous a manqué.

Notre corpus ne portait que sur les couples adjectif-nom mais la compositionnalité concerne également d'autres couples. Le corpus de similarité de Mitchell & Lapata [M&L10] nous permet d'évaluer le modèle sur des couples verbe-objet ou des noms composés (nom-nom). Il est donc possible de tester le modèle sur ces couples et de voir si les résultats sont meilleurs ou moins bons selon le type de couple étudié.

Il serait intéressant de faire des études de cas plus approfondies sur les duos de couples adjectif-nom sur la similarité desquels humains et autoencodeur ne s'entendent pas afin d'améliorer les résultats, que ce soit grâce à *WordNet* ou avec d'autres apports linguistiques, des règles par exemple.

L'objet de ce stage n'était qu'un test pour voir s'il était intéressant d'étudier ce modèle de réseau de neurones pour les tâches utilisées en évaluation avant de l'utiliser pour la représentation de la sémantique en factorisation de tenseurs. La réussite du modèle, bien que mitigée, devrait encourager la poursuite des recherches et le lancement des tests concernant directement les tenseurs.

Bibliographie

[Ash11] : Asher, N. (2011). *Lexical meaning in context: A web of words*. Cambridge University Press.

[B&C12] : Briscoe, T., & Carroll, J. A. (2002, May). Robust Accurate Statistical Annotation of General Text. In *LREC*.

[B&Z10] : Baroni, M., & Zamparelli, R. (2010, October). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1183-1193). Association for Computational Linguistics.

[BD+03] : Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, *3*, 1137-1155.

[C&W08] : Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

[CSC10] : CCoecke, B., Sadrzadeh, M., & Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

[D&L10] : Dinu, G., & Lapata, M. (2010, October). Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1162-1172). Association for Computational Linguistics.

[EPa08] : Erk, K., & Padó, S. (2008, October). A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 897-906). Association for Computational Linguistics.

[EPb09] : Erk, K., & Padó, S. (2009, March). Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics* (pp. 57-65). Association for Computational Linguistics.

[Fel98] : Fellbaum, C. (1998). *WordNet*. Blackwell Publishing Ltd.

[GA+06] : Guthrie, D., Allison, B., Liu, W., Guthrie, L., & Wilks, Y. (2006). A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)* (pp. 1-4).

[Gie10] : Giesbrecht, E. (2010, June). Towards a matrix-based distributional model of meaning. In *Proceedings of the NAACL HLT 2010 Student Research Workshop* (pp. 23-28). Association for Computational Linguistics.

[GSa11] : Grefenstette, E., & Sadrzadeh, M. (2011, July). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1394-1404). Association for Computational Linguistics.

[GSb11] : GGrefenstette, E., & Sadrzadeh, M. (2011, July). Experimenting with transitive verbs in a discocat. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics* (pp. 62-66). Association for Computational Linguistics.

[Har54] : Harris, Z. S. (1954). Distributional structure. *Word*.

[Jan01] : Janssen, T. M. (2001). Frege, contextuality and compositionality. *Journal of Logic, Language and Information*, *10*(1), 115-136.

[Les86] : Lesk, M. (1986, June). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation* (pp. 24-26). ACM.

[LT+98] : Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, *25*(2-3), 259-284.

[M&H07] : Mnih, A., & Hinton, G. (2007, June). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning* (pp. 641-648). ACM.

[M&L08] : Mitchell, J., & Lapata, M. (2008, June). Vector-based Models of Semantic Composition. In *ACL* (pp. 236-244).

[M&L10] : Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive science*, *34*(8), 1388-1429.

[MMS93] : Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, *19*(2), 313-330.

[Par95] : Partee, B. (1995). Lexical semantics and compositionality. *An invitation to cognitive science: Language*, *1*, 311-360.

[Puj91] : Pustejovsky, J. (1991). The generative lexicon. *Computational linguistics*, *17*(4), 409-441.

[S+a13] : Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems* (pp. 926-934).

[S+b13] : Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)* (Vol. 1631, p. 1642).

[San90] : Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision).

[SH+12] : Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012, July). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1201-1211). Association for Computational Linguistics.

[T&P10] : Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, *37*(1), 141-188.

[TD+13] : Tsubaki, M., Duh, K., Shimbo, M., & Matsumoto, Y. (2013). Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks. In *EMNLP* (pp. 130-140).

[TDP09] : Thater, S., Dinu, G., & Pinkal, M. (2009, August). Ranking paraphrases in context. In *Proceedings of the 2009 Workshop on Applied Textual Inference* (pp. 44-47). Association for Computational Linguistics.

[TFP10] : Thater, S., Fürstenau, H., & Pinkal, M. (2010, July). Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 948-957). Association for Computational Linguistics.

[VDC10] : Van de Cruys, T. (2010). A non-negative tensor factorization model for selectional preference induction. *Natural Language Engineering*, *16*(04), 417-437.

[VL+08] : Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103). ACM.

[VPK13] : Van de Cruys, T., Poibeau, T., & Korhonen, A. (2013). A tensor-based factorization model of semantic compositionality. In *Conference of the North American Chapter of the Association of Computational Linguistics (HTL-NAACL)* (pp. 1142-1151).

[W&K91] : Weiss, S. M., & Kulikowski, C. A. (1991). Computer systems that learn: classification and prediction methods from statistics. *Neural Networks, Machine Learning, and Expert Systems*.

[YD+13] : Yu, D., Deng, L., & Seide, F. (2013). The deep tensor neural network with applications to large vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, *21*(2), 388-396.